Code 128

Code 128 is a high-density linear barcode symbology defined in ISO/IEC 15417:2007.[1] It is used for alphanumeric or numeric-only barcodes. It can encode all 128 characters of ASCII and, by use of an extension symbol (FNC4), the Latin-1 characters defined in ISO/IEC 8859-1.. It generally results in more compact barcodes compared to other methods like Code 39, especially when the texts contain mostly digits.

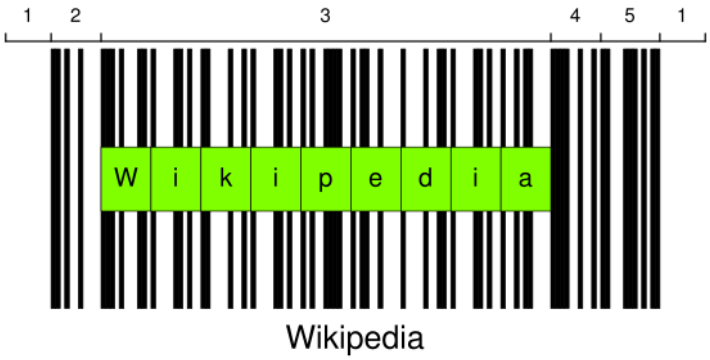
A Swiss postal barcode encoding "RI 476 394 652 CH" in Code 128-B

GS1-128 (formerly known as UCC/EAN-128) is a subset of Code 128 and is used extensively worldwide in shipping and packaging industries as a product identification code for the container and pallet levels in the supply chain.

Specification

A Code 128 barcode has seven sections:

1. Quiet zone
2. Start symbol
3. Encoded data
4. Check symbol (mandatory)
5. Stop symbol
6. Final bar (often considered part of the stop symbol)
7. Quiet zone



Schematic of a Barcode (Code 128B).
1:quiet zone, **2**:start code, **3**:data, **4**:checksum, **5**:stop code

The check symbol is calculated from a weighted sum (modulo 103) of all the symbols.

Subtypes

Code 128 includes 108 symbols: 103 data symbols, 3 start symbols, and 2 stop symbols. Each symbol consists of three black bars and three white spaces of varying widths. All widths are multiples of a basic "module". Each bar and space is 1 to 4 modules wide, and the symbols are fixed width: the sum of the widths of the three black bars and three white bars is 11 modules.

The stop pattern is composed of two overlapped symbols and has four bars. The stop pattern permits bidirectional scanning. When the stop pattern is read left-to-right (the usual case), the stop symbol (followed by a 2-module bar) is recognized. When the stop pattern is read right-to-left, the reverse stop symbol (followed by a 2-module bar) is recognized. A scanner seeing the reverse stop symbol then knows it must skip the 2-module bar and read the rest of the barcode in reverse.

Despite its name, Code 128 does not have 128 distinct symbols, so it cannot represent 128 code points directly. To represent all 128 ASCII values, it shifts among three code sets (A, B, C). Together, code sets A and B cover all 128 ASCII characters. Code set C is used to efficiently encode digit strings. The initial subset is selected by using the appropriate start symbol. Within each code set, some of the 103 data code

points are reserved for shifting to one of the other two code sets. The shifts are done using code points 98 and 99 in code sets A and B, 100 in code sets A and C and 101 in code sets B and C to switch between them):

- 128A (Code Set A) – ASCII characters 00 to 95 (0–9, A–Z and control codes), special characters, and FNC 1–4
- 128B (Code Set B) – ASCII characters 32 to 127 (0–9, A–Z, a–z), special characters, and FNC 1–4
- 128C (Code Set C) – 00–99 (encodes two digits with a single code point) and FNC1

Quiet zone

The minimum width of the Quiet Zone to the left and right of the 128 Bar Code is 10x, where x is the minimum width of a module. It is mandatory at the left and right side of the barcode.

Start/stop and encoded data

Each symbol in the barcode is composed of three bars and three spaces. Each bar or space is 1, 2, 3 or 4 units wide, the sum of the widths of bars must be even (4, 6 or 8 units), the sum of the widths of the spaces must be odd (3, 5 or 7 units), and total 11 units per symbol. For instance, encoding the ASCII character "0" can be viewed as 10011101100, where a sequence of 1's is a bar and a sequence of 0's is a space. A single 1 would be the thinnest line in the bar code. Three 1's in sequence (111) indicates a bar three times as thick as a single 1 bar.

There are 108 possible 11-unit wide symbols, and the code uses all possible symbols. Two of the symbols are used for stop (end-of-barcode) indication, stop and reverse stop. The two stop symbols are special because they are always followed by a 2-unit bar, forming a 13-unit long stop pattern. Reading the stop pattern left to right is the stop symbol (followed by a 2-unit bar), and reading the stop pattern right to left is the reverse stop symbol (followed by a 2-unit bar).

Check digit calculation

The check digit is a weighted modulo-103 checksum. It is calculated by summing the start code 'value' to the products of each symbol's 'value' multiplied by its position in the barcode string. The start symbol *and* first encoded symbol are in position 1. The sum of the products is then reduced modulo 103. The remainder is then converted back to one of the 103 non-delimiter symbols (following the instructions given below) and appended to the barcode, immediately before the stop symbol.

For example, in the following table, the code 128 variant A checksum value is calculated for the alphanumeric string PJJ123C:

Code	Value	Position	Value × Position
Start Code A	103	1	103
P	48	1	48
J	42	2	84
J	42	3	126
1	17	4	68
2	18	5	90
3	19	6	114
C	35	7	245
Sum			878
Remainder mod 103			54

For the purpose of computing the check symbol, the shift and code-switch symbols are treated the same as any other symbol in the bar code. The checksum is computed over the symbol values, without regard for which code set is active at the time. For instance the code set C value "33" and the code set B value "A" are both considered to be a Code 128 value of 33, and the check digit is computed based on the value of 33 times the symbol's position within the barcode.

Using FNC4 to encode high (128–255) characters

The special symbol FNC4 ("Function 4"), present only in code sets A and B, can be used to encode all the Latin-1 (ISO-8859-1) characters in a Code 128 barcode.[2] The feature is not widely supported and is not used in GS1-128.[3][4] When a single FNC4 is present in a string, the following symbol is converted to ASCII as usual, and then 128 is added to the ASCII value. (If the following symbol is a shift, then a second symbol will be used to obtain the character.) If two FNC4s are used consecutively then all following characters will be treated as such, up to the end of the string or another pair of FNC4s. Between the double FNC4s, a single FNC4 will be used to denote that the following character will be standard ASCII.[5]

Bar code widths

Code128 specifies a combination of 6 alternating bars and spaces (3 of each) for each symbol. Thus, each symbol begins with a bar and ends with a space. In barcode fonts, the final bar is generally combined with the stop symbol to make a wider stop pattern. The following table details the widths associated with each bar and space for each symbol. The width of each bar or space may be 1, 2, 3 or 4 units (modules). Using the example above, an 'A' would be depicted with the pattern 10100011000, or as widths 111323 in the tables below.

The widths value is derived by counting the length of each run of 1's then 0's in the pattern, starting from the left. There will always be 6 runs and the lengths of these 6 runs form the Widths value. For example, using the pattern 10100011000, the run lengths are 1 (digit 1), 1 (digit 0), 1 (digit 1), 3 (digit 0), 2 (digit 1), 3 (digit 0). Reporting just the lengths of each run gives 1, 1, 1, 3, 2, 3, thereby producing a widths value of 111323.

Code 128

Value	Hex Value	128A	128B	128C	Font position (Common/Uncommon/Barcodesoft)		Bar/Space	
					Code	Latin-1	Pattern	Widths
0	00	space	space	00	32 or 194 or 207 / 212 / 252	‗ or Â or Ï / Ô / ü	11011001100	212222
1	01	!	!	01	33	!	11001101100	222122
2	02	"	"	02	34	"	11001100110	222221
3	03	#	#	03	35	#	10010011000	121223
4	04	$	$	04	36	$	10010001100	121322
5	05	%	%	05	37	%	10001001100	131222
6	06	&	&	06	38	&	10011001000	122213
7	07	'	'	07	39	'	10011000100	122312
8	08	(	(	08	40	(	10001100100	132212
9	09	)	)	09	41	)	11001001000	221213
10	0a	*	*	10	42	*	11001000100	221312
11	0b	+	+	11	43	+	11000100100	231212
12	0c	,	,	12	44	,	10110011100	112232
13	0d	-	-	13	45	-	10011011100	122132
14	0e	.	.	14	46	.	10011001110	122231
15	0f	/	/	15	47	/	10111001100	113222
16	10	0	0	16	48	0	10011101100	123122
17	11	1	1	17	49	1	10011100110	123221
18	12	2	2	18	50	2	11001110010	223211
19	13	3	3	19	51	3	11001011100	221132
20	14	4	4	20	52	4	11001001110	221231
21	15	5	5	21	53	5	11011100100	213212
22	16	6	6	22	54	6	11001110100	223112
23	17	7	7	23	55	7	11101101110	312131
24	18	8	8	24	56	8	11101001100	311222
25	19	9	9	25	57	9	11100101100	321122
26	1a	:	:	26	58	:	11100100110	321221
27	1b	;	;	27	59	;	11101100100	312212
28	1c	<	<	28	60	<	11100110100	322112
29	1d	=	=	29	61	=	11100110010	322211
30	1e	>	>	30	62	>	11011011000	212123
31	1f	?	?	31	63	?	11011000110	212321
32	20	@	@	32	64	@	11000110110	232121
33	21	A	A	33	65	A	10100011000	111323
34	22	B	B	34	66	B	10001011000	131123
35	23	C	C	35	67	C	10001000110	131321
36	24	D	D	36	68	D	10110001000	112313

Value	Hex Value	128A	128B	128C	Font position (Common/Uncommon/Barcodesoft)		Bar/Space	
					Code	Latin-1	Pattern	Widths
37	25	E	E	37	69	E	10001101000	132113
38	26	F	F	38	70	F	10001100010	132311
39	27	G	G	39	71	G	11010001000	211313
40	28	H	H	40	72	H	11000101000	231113
41	29	I	I	41	73	I	11000100010	231311
42	2a	J	J	42	74	J	10110111000	112133
43	2b	K	K	43	75	K	10110001110	112331
44	2c	L	L	44	76	L	10001101110	132131
45	2d	M	M	45	77	M	10111011000	113123
46	2e	N	N	46	78	N	10111000110	113321
47	2f	O	O	47	79	O	10001110110	133121
48	30	P	P	48	80	P	11101110110	313121
49	31	Q	Q	49	81	Q	11010001110	211331
50	32	R	R	50	82	R	11000101110	231131
51	33	S	S	51	83	S	11011101000	213113
52	34	T	T	52	84	T	11011100010	213311
53	35	U	U	53	85	U	11011101110	213131
54	36	V	V	54	86	V	11101011000	311123
55	37	W	W	55	87	W	11101000110	311321
56	38	X	X	56	88	X	11100010110	331121
57	39	Y	Y	57	89	Y	11101101000	312113
58	3a	Z	Z	58	90	Z	11101100010	312311
59	3b	[	[	59	91	[	11100011010	332111
60	3c	\	\	60	92	\	11101111010	314111
61	3d	]	]	61	93	]	11001000010	221411
62	3e	^	^	62	94	^	11110001010	431111
63	3f	_	_	63	95	_	10100110000	111224
64	40	NUL	`	64	96	`	10100001100	111422
65	41	SOH	a	65	97	a	10010110000	121124
66	42	STX	b	66	98	b	10010000110	121421
67	43	ETX	c	67	99	c	10000101100	141122
68	44	EOT	d	68	100	d	10000100110	141221
69	45	ENQ	e	69	101	e	10110010000	112214
70	46	ACK	f	70	102	f	10110000100	112412
71	47	BEL	g	71	103	g	10011010000	122114
72	48	BS	h	72	104	h	10011000010	122411
73	49	HT	i	73	105	i	10000110100	142112
74	4a	LF	j	74	106	j	10000110010	142211
75	4b	VT	k	75	107	k	11000010010	241211

Value	Hex Value	128A	128B	128C	Font position (Common/Uncommon/Barcodesoft)		Bar/Space	
					Code	Latin-1	Pattern	Widths
76	4c	FF	l	76	108	l	11001010000	221114
77	4d	CR	m	77	109	m	11110111010	413111
78	4e	SO	n	78	110	n	11000010100	241112
79	4f	SI	o	79	111	o	10001111010	134111
80	50	DLE	p	80	112	p	10100111100	111242
81	51	DC1	q	81	113	q	10010111100	121142
82	52	DC2	r	82	114	r	10010011110	121241
83	53	DC3	s	83	115	s	10111100100	114212
84	54	DC4	t	84	116	t	10011110100	124112
85	55	NAK	u	85	117	u	10011110010	124211
86	56	SYN	v	86	118	v	11110100100	411212
87	57	ETB	w	87	119	w	11110010100	421112
88	58	CAN	x	88	120	x	11110010010	421211
89	59	EM	y	89	121	y	11011011110	212141
90	5a	SUB	z	90	122	z	11011110110	214121
91	5b	ESC	{	91	123	{	11110110110	412121
92	5c	FS	|	92	124	|	10101111000	111143
93	5d	GS	}	93	125	}	10100011110	111341
94	5e	RS	~	94	126	~	10001011110	131141
95	5f	US	DEL	95	195 / 200 / 240	Ã / È / ð	10111101000	114113
96	60	FNC 3	FNC 3	96	196 / 201 / 241	Ä / É / ñ	10111100010	114311
97	61	FNC 2	FNC 2	97	197 / 202 / 242	Å / Ê / ò	11110101000	411113
98	62	Shift B	Shift A	98	198 / 203 / 243	Æ / Ë / ó	11110100010	411311
99	63	Code C	Code C	99	199 / 204 / 244	Ç / Ì / ô	10111011110	113141
100	64	Code B	FNC 4	Code B	200 / 205 / 245	È / Í / õ	10111101110	114131
101	65	FNC 4	Code A	Code A	201 / 206 / 246	É / Î / ö	11101011110	311141
102	66	FNC 1	FNC 1	FNC 1	202 / 207 / 247	Ê / Ï / ÷	11110101110	411131
103	67	Start Code A			203 / 208 / 248	Ë / Ð / ø	11010000100	211412
104	68	Start Code B			204 / 209 / 249	Ì / Ñ / ù	11010010000	211214
105	69	Start Code C			205 / 210 / 250	Í / Ò / ú	11010011100	211232
106	6a	Stop			—	—	11000111010	233111
—	—	Reverse Stop			—	—	11010111000	211133
—	—	Stop pattern (7 bars/spaces)			206 / 211 / 251	Î / Ó / û	1100011101011	2331112

The "Code A", "Code B" and "Code C" symbols cause all following symbols to be interpreted according to the corresponding subcode (i.e. 128A, 128B or 128C). The "Shift" symbol switches a single following symbol's interpretation between subcodes A and B.

The encoded ASCII char depends on the actual used barcode-font. Especially the ASCII char of value 0 and of value 95 and above may be defined differently in the font that is installed.

The FNC*x* codes are used for special purposes. FNC1 at the beginning of a bar code indicates a GS1-128 bar code which begins with a 2- 3- or 4-digit *application identifier* assigned by the Uniform Code Council, which explains the following digits. For example, application identifier 421 indicates that an ISO 3166-1 numeric country code and ship-to postal code follows. Thus, the U.S. ZIP code for the White House would generally be printed as "(421) 840 20500", but would actually be coded as "`[Start C] [FNC1] 42 18 40 20 50 [Code A] 16 [Check symbol 92] [Stop]`"

Check digit calculation for the above Zip code example:

	Value	Weight	Weight × Value
Start C	105	1	105
FNC1	102	1	102
42	42	2	84
18	18	3	54
40	40	4	160
20	20	5	100
50	50	6	300
Code A	101	7	707
0	16	8	128
		Sum =	1740
1740	Mod 103 =	92	

Availability

For the end user, Code 128 barcodes may be generated by either an outside application to create an image of the barcode, or by a font-based barcode solution. Either solution requires the use of an application or an application add in to calculate the check digit and create the barcode.

Barcode length optimization

Code set C uses one code symbol to represent two digits, so when the text contains just digits it will generally result in shorter barcodes. However, when the string contains only a few digits or it's mixed with non-digit character, it does not always produce a more compact code than code sets A or B. Using code set C saves one symbol per two digits, but costs a mode-shift symbol to enter and exit the set. Thus, it is only worth using if there are enough consecutive digits. For example, encoding the string "X00Y" with code set A or B requires 7 code symbols (`[Start B] 56 16 16 57 [checksum] [Stop]`), while using code set C for the "00" would result in a code 8 symbols long (`[Start B] 56 [Code C] 00 [Code B] 57 [checksum] [Stop]`).

Using code set C is only advantageous under the following conditions:

Location of digits	Number of consecutive digits
beginning of data	4+
end of data	4+
middle of data (surrounded by symbols from code set A or B)	6+
entire data	either 2 or 4+ (but not 3)

At the end of a string, delaying the transition to code set C until there are an even number of digits remaining avoids an extra symbol. Consider the string "...01234": a delayed switch produces `... 0 [Code C] 12 34 [checksum] [Stop]` but an early switch produces `... [Code C] 01 23 [Code A]`

`4 [checksum] [Stop].`[6]

For example, given the string "098x1234567y23", savings on barcode length using code set C are achieved only if it is applied to middle part of the string. For the beginning and ending part of the string, switching to code set C is not effective. As there are an odd number of digits in the middle of the string, the odd one must be use a different code, set, but it makes no difference whether this is the first or last; 16 symbols are required in either case: `[Start B] 0 9 8 x 1 [Code C] 23 45 67 [Code B] y 2 3 [checksum] [Stop]`, or `[Start B] 0 9 8 x [Code C] 12 34 56 [Code B] 7 y 2 3 [checksum] [Stop]`.

Optimizing the length of the resulting barcode is important when barcode readers are used which must detect the entire barcode image at once in order to read it, such as common laser scanners. The longer the barcode is, the greater distance of laser barcode reader from barcode image is needed, making reading difficult or impossible above some threshold lengths/distances.

The optimal encoding can be found using a dynamic programming algorithm.[7]

References

1. "ISO/IEC 15417:2007 - Information technology -- Automatic identification and data capture techniques -- Code 128 bar code symbology specification" (https://www.iso.org/standard/43896.html). *www.iso.org*. Retrieved 2018-02-15.
2. Apparently ISO 15417 Annex F
3. "Code 128 Explained" (http://www.softmatic.com/barcode-code-128.html). Softmatic GmbH. Retrieved 2017-01-21. "In principle non-ASCII characters like German umlauts (e.g. ÄÖÜ) can be encoded in a Code 128 symbol by using a special character (FNC4). However, this feature is not widely supported. Using a 2D barcode symbology like Aztec or Datamatrix with dedicated support for non-ASCII data might be a better choice."
4. GS1 General Specifications (January 2006 – Version 7.0), section 5.3.1.1 GS1-128 Symbology Characteristics, stating, "Characters with ASCII values 128 to 255 may also be encoded in Code 128 Symbols. Characters with ASCII values 128 to 255 accessed by Function 4 Character (FNC4) are reserved for future use and are not used in GS1-128 Bar Code Symbols."
5. "TBarcode1D_Code128" (http://www.han-soft.com/releases/barcode1d/documents/b_code128.html). Han-soft corporation. Retrieved 2017-01-21. "If a single "FNC 4" character is used, indicates the following data character in the symbol is an extended ASCII character. A 'SHIFT' character may follow the 'FNC 4' character if it is necessary to change character subset for the following data character. Subsequent data characters revert to the standard ASCII character set. If two consecutive 'FNC4' characters are used, all following data characters are extended ASCII characters until two further consecutive 'FNC4' characters are encountered or the end of the symbol is reached. If during this sequence of extended encodation a single "FNC4" character is encountered it is used to revert to standard ASCII encodation for the next data character only. 'SHIFT' and character subset characters shall have their normal effect during such a sequence."
6. GS1 General Specifications, Version 13, Issue 1, Jan-2013, Section 5.4.7.7. Use of Start, Code Set, and Shift symbols to Minimize Symbol Length (Informative), pages 268 to 269. This section gives the compression strategy.
7. Skiena, Steven S. (2010). "8.9 War Story: Text Compression for Bar Codes". *The Algorithm Design Manual* (2nd ed.). ISBN 1-849-96720-2. "dynamic programming led to an 8% tighter encoding on average."

External links

- GS1-128 Specification (http://mdn.morovia.com/kb/GS1-128-Specification-10625.html) – A detailed list of Application Identifiers.
- Barcodesoft (https://web.archive.org/web/20121016032219/http://barcodesoft.com/code128mapping.htm) – Font mapping of Barcodesoft, which differs from the common ascii mapping (see http://ascii-code.com/).

- The 128 code (http://grandzebu.net/informatique/codbar-en/code128.htm) – Learn the Code 128 encoding algorithm with a font-based barcode solution.
- Online barcode generator (http://barcode.tec-it.com) – Free online Barcode generator for the various barcode types.

Sample code

- ZXing (https://github.com/zxing/zxing) – Multiplatform open source barcode scanner / generator with versions available in Java (core project) and ports to ActionScript, C++, C#, ObjectiveC and Ruby.
- Python Bar Code 128 (http://barcode128.blogspot.com) – This code appears to draw boxes one pixel wide. It appears it was modified from a short line long line bar code which would have drawn lines. The "Black boxes" should be the same size as the "White Boxes".
- GenCode128 (http://www.codeproject.com/KB/GDI-plus/GenCode128.aspx) – Free C# source code implementation of Code128. Almost all features are implemented, but is not 100% complete.
- Barcode1DTools Ruby gem (https://github.com/mdchaney/Barcode1DTools) – Ruby source code for many 1D barcode symbologies including Code 128.
- Perl barcode generation code (https://github.com/mdchaney/barcodes) – Perl source code for many 1D barcode symbologies including Code 128.
- Barcode::Code128 (https://metacpan.org/module/Barcode::Code128/) – Free Perl barcode generation module.
- GOCR (http://jocr.sourceforge.net/index.html) – Free OCR with Code 128 recognition.
- Barcode Code 128 (http://zanstra.com/my/Barcode.html) – Free JavaScript source code implementation of Code128.
- Barcode4J (http://barcode4j.sourceforge.net/index.html) – Free Java API with implementation of Code128 and other standard barcodes.
- JavaScript Code 128 (http://www.barcoderesource.com/htmlBarcode.shtml) – Open-source JavaScript implementation of Code128 and other linear barcodes.
- Introducing creation of Code 128 barcodes (http://sheepdogguides.com/lut/lt4Nf-Barcode.htm) Guide to converting text to Code 128 barcodes. Written for Lazarus (open-source, multi-platform GUI Pascal) but of general use.
